regsub manual page - Tcl Built-In Commands

tcl.tk/man/tcl/TclCmd/regsub.htm

NAME

regsub — Perform substitutions based on regular expression pattern matching

SYNOPSIS

regsub ?*switches*? *exp string subSpec* ?*varName*?

DESCRIPTION

This command matches the regular expression *exp* against *string*, and either copies *string* to the variable whose name is given by *varName* or returns *string* if *varName* is not present. (Regular expression matching is described in the **re_syntax** reference page.) If there is a match, then while copying *string* to *varName* (or to the result of this command if *varName* is not present) the portion of *string* that matched *exp* is replaced with *subSpec*. If *subSpec* contains a “&” or “\0”, then it is replaced in the substitution with the portion of *string* that matched *exp*. If *subSpec* contains a “\n”, where *n* is a digit between 1 and 9, then it is replaced in the substitution with the portion of *string* that matched the *n*'th parenthesized subexpression of *exp*. Additional backslashes may be used in *subSpec* to prevent special interpretation of “&”, “\0”, “\n” and backslashes. The use of backslashes in *subSpec* tends to interact badly with the Tcl parser's use of backslashes, so it is generally safest to enclose *subSpec* in braces if it includes backslashes.
If the initial arguments to **regsub** start with **-** then they are treated as switches. The following switches are currently supported:

-all
All ranges in *string* that match *exp* are found and substitution is performed for each of these ranges. Without this switch only the first matching range is found and substituted. If **-all** is specified, then “&” and “\n” sequences are handled for each substitution using the information from the corresponding match.

-expanded
Enables use of the expanded regular expression syntax where whitespace and comments are ignored. This is the same as specifying the **(?x)** embedded option (see the **re_syntax** manual page).

-line
Enables newline-sensitive matching. By default, newline is a completely ordinary character with no special meaning. With this flag, “[^” bracket expressions and “.” never match newline, “^” matches an empty string after any newline in addition to its normal function, and “$” matches an empty string before any newline in addition to its normal function. This flag is equivalent to specifying both **-linestop** and **-lineanchor**, or the **(?n)** embedded option (see the **re_syntax** manual page).

-linestop
Changes the behavior of “[^” bracket expressions and “.” so that they stop at newlines. This is the same as specifying the **(?p)** embedded option (see the **re_syntax** manual page).

-lineanchor
Changes the behavior of “^” and “$” (the “anchors”) so they match the beginning and end of a line respectively. This is the same as specifying the **(?w)** embedded option (see the **re_syntax** manual page).

-nocase
Upper-case characters in *string* will be converted to lower-case before matching against *exp*; however, substitutions specified by *subSpec* use the original unconverted form of *string*.

-start *index*
Specifies a character index offset into the string to start matching the regular expression at. The *index* value is interpreted in the same manner as the *index* argument to **string index**. When using this switch, “^” will not match the beginning of the line, and \A will still match the start of the string at *index*. *index* will be constrained to the bounds of the input string.

--
Marks the end of switches. The argument following this one will be treated as *exp* even if it starts with a **-**.

If *varName* is supplied, the command returns a count of the number of matching ranges that were found and replaced, otherwise the string after replacement is returned. See the manual entry for **regexp** for details on the interpretation of regular expressions.

EXAMPLES

Replace (in the string in variable *string*) every instance of **foo** which is a word by itself with **bar**:

```
regsub -all {\mfoo\M} $string bar string
```

or (using the “basic regular expression” syntax):

```
regsub -all {(?b)\<foo\>} $string bar string
```

Insert double-quotes around the first instance of the word **interesting**, however it is capitalized.

```
regsub -nocase {\yinteresting\y} $string {"&"} string
```

Convert all non-ASCII and Tcl-significant characters into \u escape sequences by using **regsub** and **subst** in combination:

```
# This RE is just a character class for almost everything "bad"
set RE {[][{};#\\\$ \r\t\u0080-\uffff]}

# We will substitute with a fragment of Tcl script in brackets
set substitution {[format \\\\u%04x [scan "\\&" %c]]}

# Now we apply the substitution to get a subst-string that
# will perform the computational parts of the conversion. Note
# that newline is handled specially through string map since
# backslash-newline is a special sequence.
set quoted [subst [string map {\n {\\u000a}} \
        [regsub -all $RE $string $substitution]]]
```